Topi Kahila

IoT-laitteen turvallinen tiedonsiirto pilvipalveluun käyttäen NB-IoT-verkkoteknologiaa

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

3.5.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Topi Kahila IoT-laitteen turvallinen tiedonsiirto pilvipalveluun käyttäen NB-IoT-verkkoteknologiaa 32 sivua 3.5.2018
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Smart Systems
Ohjaajat	Lehtori Kimmo Sauren Toimitusjohtaja Jussi Heiskanen

Insinöörityön tarkoituksena oli tutkia esineiden internetin tietoturvallisuutta ja salausmenetelmiä. Tämän lisäksi rakennettiin IoT-prototyyppi, jolla voitiin demonstroida turvallista tiedonsiirtoa IoT-laitteelta pilvipalveluun hyödyntäen Narrowband IoT -verkkoteknologiaa. Nykyiset salausmenetelmät ovat raskaita toteuttaa vähävirtaisilta IoT-laitteilta, joten työn tarkoituksena oli myös selvittää turvallisimmat ja energiatehokkaimmat tavat salata liikennettä.

Käytännön toteutus aloitettiin prototyypin perusyhteyksien luomisella. Tämän jälkeen käyttöön otettiin MQTT-tiedonsiirtoprotokolla sekä Google IoT Coren vaatima käyttäjän varmennemenetelmä. Ongelmaksi toteutuksen loppuvaiheessa muodostui se, että kehitysalustasta puuttui kokonaan Google IoT Core -pilvipalvelun vaatima suojauspaketin tuki.

Työn lopputuloksena onnistuttiin luomaan toimiva MQTT-protokolla ja JSON web token -käyttäjävarmenne. Insinöörityön puitteissa prototyypin ja pilvipalvelun yhteyttä ei saatu suojauspaketin tuen puuttuessa toimimaan. Suojauspaketin tuki lisätään kehitysalustaan lähitulevaisuudessa päivityksen mukana, ja tuolloin prototyypillä pystytään havainnollistamaan tuotteen toimintaa ja tiedonsiirron turvallisuutta.

Avainsanat	Narrowband, esineiden internet, MQTT, tietoturva

Author Title Number of Pages Date	Topi Kahila Secured connection from IoT-device to cloud service using NB-IoT radio technology 32 pages 3 May 2018
Degree	Bachelor of Engineering
Degree Programme	Information and Communication Technology
Professional Major	Smart Systems
Instructors	Kimmo Sauren, Senior Lecturer Jussi Heiskanen, CEO

The goal of this thesis was to study the security and encryption methods of the devices used in internet of things. An IoT prototype was created so that secure data transfer from IoT device into the cloud by using Narrowband IoT networking technology could be demonstrated. The current encryption methods are burdensome to be used in low powered IoT devices. Additionally, the purpose for this thesis was to clarify the most secure and energy efficient ways to conceal data traffic.

The practical implementation was started with the basic communications of the prototype. After they were functional, the required certificate system of the MQTT-communication protocol and Google's IoT Core was implemented. There was complication in the development platform about the support of required cipher suite in the final stages of the prototype.

The result of the thesis included successful usage of the MQTT protocol and JSON web token user certificates. Unfortunately, the connection between the prototype and the cloud services could not be completed due to lack of support with the cipher suite. The support for this cipher suite will be added to the development platform by the developer in the near future and then the functionality of the prototype and data transfer can be confirmed.

Keywords	Narrowband, IoT, MQTT, Information security

Sisällys

Lyhenteet

AES	Advanced Encryption Standard. Lohkosalausmenetelmä tietotekniikassa.
IoT	Internet of Things. Esineiden internet, myös teollinen internet.
JSON	JavaScript Object Notation. Yksinkertainen avoimen standardin tiedostomuoto tiedonvälitykseen.
MQTT	Message Queuing Telemetry Transport. Tiedonsiirtoprotokolla.
OEM	Original equipment manufacturer. Alkuperäinen laitevalmistaja.
TLS	Transport Layer Security. Salausprotokolla, jolla suojataan internetsovellusten tietoliikenne.
VPN	Virtual Private Network. Virtuaalinen erillisverkko.
XML	Extensible Markup Language. Standardi, jolla tiedon merkitys on kuvattavissa tiedon sekaan.

1 Johdanto

Esineiden internet on kasvanut nopeasti, ja se kasvaa edelleen. Se koostuu tyypillisesti pienistä vähävirtaisista sensoreista, jotka välittävät dataa ympäristöstään. Tällaisia internetiin kytkettyjä asioita ja esineitä kutsutaan IoT-laitteiksi. Usein IoT-laitteet ovat mahdollisimman halvalla tuotettuja, ja usein rahaa on säästetty juuri tietoturvan osalta. Nykyaikaiset salausmenetelmät vaativat paljon laskentatehoa, joka kuluttaa langattomasti toimivan IoT-laitteen käyttöaikaa, mikä osaltaan laskee IoT-kehittäjien mielenkiintoa turvallisuuteen varsinkin, kun alalla kilpaillaan usein laitteen pariston kestolla.

Insinöörityön tavoitteena on suunnitella ja toteuttaa IoT-laitteen prototyyppi asiakasyritykselle nimeltä Wizense. Se on vuonna 2016 perustettu yritys, joka on erikoistunut tarjoamaan teollisen internetin ratkaisuja. Tässä insinöörityöraportissa käsitellään tietoturvallisuutta esineiden internetissä, erilaisia salausmenetelmiä sekä tutustutaan Narrowband IoT -verkkoteknologiaan.

Prototyypin tarkoituksena on luoda kevyt mutta turvallinen tiedonsiirto IoT-laitteelta Google IoT Core -pilvipalveluun käyttäen MQTT-tiedonsiirtoprotokollaa ja Narrowband IoT -verkkoteknologiaa. Ohjelma on tarkoitus siirtää myöhemmin yrityksen kehittämälle IoT-laitteelle. Prototyyppi toteutetaan Nordic Semiconductor NRF52- ja u-blox SARA-R4 -kehitysalustoilla, joissa olevia teknologioita käytetään lopputuotteessa.

2 Tietoturva ja sen haasteet esineiden internetissä

2.1 Tietoturva yleisesti

Tietoturvalla tarkoitetaan yleisesti toimenpiteitä, joilla pyritään takaamaan ohjelmistojen, laitteistojen ja tietoaineistojen luottamuksellisuus, käytettävyys ja eheys. Tietoturva koskettaa niin yrityksiä kuin yksityishenkilöitäkin. [1.] Yksityishenkilön yksityisyys on uhattuna erilaisten huijausyritysten, kiristyshaittaohjelmien ja tilausansojen vuoksi. Saman salasanan käyttäminen useassa eri verkkopalvelussa on myös yksittäinen iso riskitekijä internetissä. [2.]

Yrityksillä on ollut aikaisemminkin vastuu käsittelemästään datasta, mutta uuden eurooppalaisen tietosuoja-asetuksen myötä tietoturvaan tulee kiinnittää entistä enemmän huomiota. Eurooppalainen tietosuoja-asetus astuu voimaan toukokuussa 2018, ja siinä linjataan, että yritysten tulee lisätä sisäistä valvontaansa ja säännöllisesti arvioida omien toimenpiteidensä tehokkuutta tietojenkäsittelyturvallisuuden takaamiseksi. Tietosuoja-asetusten laiminlyönnistä yritykselle voidaan määrätä sanktiota enimmillään 20 miljoonaa euroa tai 4 % yrityksen vuotuisesta kokonaisliikevaihdosta. [3.]

Viestintävirasto on verkkosivuillaan listannut tietoturvalle tavoitteita, joita ovat yksilön tai organisaation annettujen tietojen eheys, luottamuksellisuus, kiistämättömyys, pääsynvalvonta, tarkastettavuus ja saatavuus. Eheydellä tarkoitetaan tiedon muuttumattomuutta sen luomis-, käsittely- tai siirtohetkellä. Kiistämättömyydellä taataan, ettei kenelläkään ole mahdollista käsitellä tietoja huomaamatta, ja pääsynvalvonnalla puolestaan, että tietoon käsiksi pääsyä rajoitetaan ja valvotaan. Saatavuudella taataan tiedon viiveetön ja helppo käyttö niille, joilla kyseiseen tietoon on oikeus. Tarkastettavuudella tarkoitetaan, että tiedon oikeellisuus tulee pystyä osoittamaan tietojenkäsittelyn jälkeen. [4.]

2.2 Esineiden internet

Esineiden internetillä, englanniksi Internet of Things (IoT, myös teollinen internet), viitataan yleensä laitteisiin ja esineisiin, jotka ovat yhdistettynä internetiin. Tyypillinen IoT-ratkaisu kerää anturien avulla tietoa itsestään ja ympäristöstään ja välittää tämän tiedon järjestelmään, jossa sitä voidaan analysoida ja hyödyntää halutulla tavalla. [5.]

Hyvä sovellusesimerkki esineiden internetistä on Suomen edustalla olevat merimerkit. Vuonna 2017 noin 600 merenkulun turvalaitetta oli jo yhteydessä internetiin, ja näin saatiin tietoa muun muassa poijujen sijainneista ja valolaitteiden tilasta. Reaaliaikainen tieto auttaa ennakoimaan vikatilanteita, lyhentämään korjausaikoja ja täten parantamanaan merenkulun turvallisuutta. [5.]

Asioiden ja esineiden internet kasvaa maailmalla kiihtyvää vauhtia, sillä verkossa käytettävien laitteiden määrä kasvoi peräti 32 prosenttia vuodesta 2015 vuoteen 2017 mennessä, jolloin laitteita oli kytkettynä internetiin 20 miljardia kappaletta. On ennustettu, että vuoteen 2020 mennessä esineitä olisi liitetty internetiin noin 31 miljardia kappaletta. [6.]

IoT on laaja käsite, joka kattaa useita toisiinsa liittyviä termejä ja tekniikoita. Nämä käsitteet voidaan yleisesti jaotella asioiden internetiksi, teolliseksi internetiksi ja esineiden internetiksi. Asioiden internetissä yksittäisellä tavaralla, esimerkiksi kirjalla, on oma identiteettinsä, vaikkeivat ne olisivatkaan kovinkaan älykkäitä. Teollisella internetillä viitataan älykkäisiin laitteisiin, jotka keräävät ja välittävät dataa laitteen tilasta ja käyttöolosuhteista hyödyntäen erityyppisiä laitteeseen kiinnitettyjä sensoreita. Teollisen internetin tavoitteena on tehostaa yrityksen liiketoimintaa. Esineiden internetiin kuuluu esineitä, jotka pystyvät kommunikoimaa internetin välityksellä joko keskenään tai pilvisovellusten kautta. [7.]

2.3 Esineiden internet ja tietoturva

IoT-laitteista puuttuu usein näyttö ja näppäimistö, mutta harva ymmärtää, että nämä laitteet tarvitsevat yhtä paljon suojausta hyökkäyksiä vastaan, kuin mikä tahansa muu laite, joka on internetissä. Yksi suurimmista IoT-laitteiden yleistymisen aiheuttamista ongelmista on se, ettei niiden suunnitteluvaiheessa ole panostettu riittävästi tietoturvaan. Tietoturvaa parantavia päivityksiä tai ohjelmistopalomuureja ei ole saatavilla, eikä edes pienillä valmistajilla ole laitteen myynnin jälkeen kiinnostusta tarjota tietoturvapäivityksiä. Tästä syystä IoT-laite on usein yksi suurimmista yksittäisistä riskitekijöistä tietoturvan kannalta. Tietoturvan laiminlyönti esimerkiksi startup-yrityksissä saattaa pahimmillaan ennen tuotteen läpilyöntiä johtaa rahoittajien perääntymiseen ja yrityksen kaatumiseen. [8.]

Jos IoT-laite on joutunut hyökkäyksen tai väärinkäytön kohteeksi, voi hyökkäystä olla erittäin vaikea havaita. Vaikka itse hyökkäyksen kohteeksi joutunut laite jatkaa toimintaansa kuten ennenkin, se voi avata hyökkääjälle pääsyn samassa sisäverkossa oleviin laitteisiin. Tällöin myös saastuneeseen IoT-laitteeseen yhteydessä olevat laitteet voivat joutua hyökkäyksen kohteeksi, vaikka ne muuten olisivat asianmukaisesti suojattuja. [8.]

IoT-laitteen suojaaminen vaatii hieman vaivaa, mutta se on kuitenkin mahdollista toteuttaa. Jotta liikkuva data ei pääse vääriin käsiin tulee laitteen yhteyksien tukea monivaiheista tunnistautumista. Kuten kaiken internetissä tapahtuvassa tiedonsiirrossa, tulee myös IoT-laitteiden tiedonsiirron hyödyntää nykyaikaista teknologiaa, kuten AES- ja TLS-salausprotokollia. Laitteiden tietoturva ja ohjelmistopäivitykset tulisi pitää päivitettyinä, sillä niissä korjataan ilmeneviä tietoturva-aukkoja. Koska päivityksiä tulee jatkuvasti, ne tulisi toteuttaa automaattisesti, jotta inhimilliset virheet voidaan minimoida. [9.]

Erityisesti edullisissa laitteissa tietoturvasta on usein tingitty, ja tällöin laite kannattaa liittää erillisen VPN-yhdyskäytävälaitteen taakse. VPN-yhdyskäytävä tarjoaa yhteyden, joka on vahvasti salattu ja todennettu, jolloin suojaamattomat laitteet voivat lähettää dataa esimerkiksi pilveen turvallisesti. [9.]

2.4 Tiedon salaaminen internetissä

Internetissä kulkeva tieto kahden osapuolen välillä kulkee useamman solmukohdan lävitse. Tähän liikenteeseen käsiksi pääsevä pystyy lukemaan kaiken tiedon, mitä sen lävitse kulkee. Jos tietoa ei salata millään tavalla, voi ulkopuolinen henkilö salakuunnella viestejä tai muokata niitä haluamallaan tavalla jälkiä jättämättä, jolloin vastaanottaja saa erilaisen viestin, kuin lähettäjä on alun perin lähettänyt. Tämän estämiseksi tieto salataan käyttämällä salausalgoritmeja. Viestit voidaan lukea salauksen läpi, mutta kaapattu sisältö olisi tässä tapauksessa lukukelvotonta. [10.]

Salausmenetelmien tehtävänä on säilyttää tieto luottamuksellisena, eheänä ja kiistämättömänä. Tavoitteena on muuttaa tieto sellaiseen muotoon, että ainoastaan se, joka tietää, miten salaus puretaan, voi lukea viestin sisällön. Toisin sanoen viesti käsitellään halutulla salausalgoritmilla käyttäen salausavainta, siten että viesti ei ole luettavissa ilman tietoa käytetystä algoritmista ja avaimesta. Perimmäinen tarkoitus on kehittää lähe-

tettävälle viestille salaus, jonka purkaminen lyhyessä ajassa ja kohtuullisella laskentateholla on mahdotonta. Vahvoja salausalgoritmeja ovat algoritmit, joita tehokkaimmatkaan tietokoneet eivät kykene murtamaan tai murtamisessa menee erittäin kauan aikaa. Tämän päivän tietokoneilla tehokkaan salauksen luominen on nopeaa laskentakapasiteetin ansiosta. Oikein toteutetulla salausalgoritmilla salattu viesti voidaan purkaa ainoastaan käymällä lävitse kaikki mahdolliset avaimet. [11; 12.]

On olemassa monia erilaisia salausmenetelmiä, mutta esittelen työssäni salaisen avaimen ja julkisen avaimet menetelmät. Salaisen avaimen menetelmät ovat symmetrisiä, ja julkisen avaimen menetelmät ovat epäsymmetrisiä.

Helsingin yliopiston [12] määritelmät salaustekniikoissa käytetyille termeille:

- Avain on merkkijono, jota käytetään sopivan salausalgoritmin kanssa tietosisällön salaamiseen ja/tai salauksen purkamiseen.

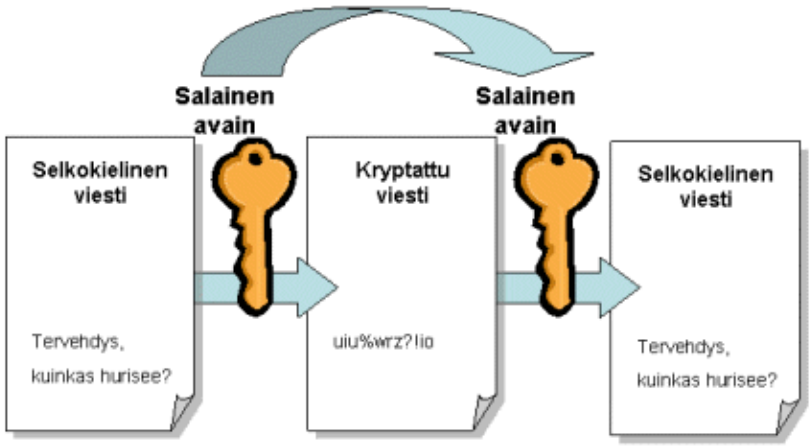
- Salainen avain on symmetriseen salaukseen ja sen purkamiseen käytetty avain.

- Yksityinen avain on julkisen avaimen menetelmässä salauksen purkamiseen ja sähköiseen allekirjoittamiseen käytetty puolisko avainparista.

- Julkinen avain on julkisen avaimen menetelmässä tiedon salaamiseen ja sähköisen allekirjoituksen tarkistamiseen käytetty puolisko avainparista.

Symmetrinen salausmenetelmä

Symmetrisessä salausmenetelmässä viesti salataan ja avataan käyttäen samaa avainta. Menetelmä on nopea ja toimii monta kertaa nopeammin kuin epäsymmetrinen salausmenetelmä, joten sillä voidaan suojata suuriakin viestejä ilman tiedonsiirtonopeuden alenemista. Menetelmässä ongelmaksi muodostuu avaimen turvallinen lähetys kaikille osapuolille ilman vaaraa, että avain päätyy väärälle taholle, sillä kuka tahansa, joka saa avaimen haltuunsa, voi myös purkaa viestin. [13.] Kuva 1 havainnollistaa symmetrisen salausmenetelmän toimintaa.

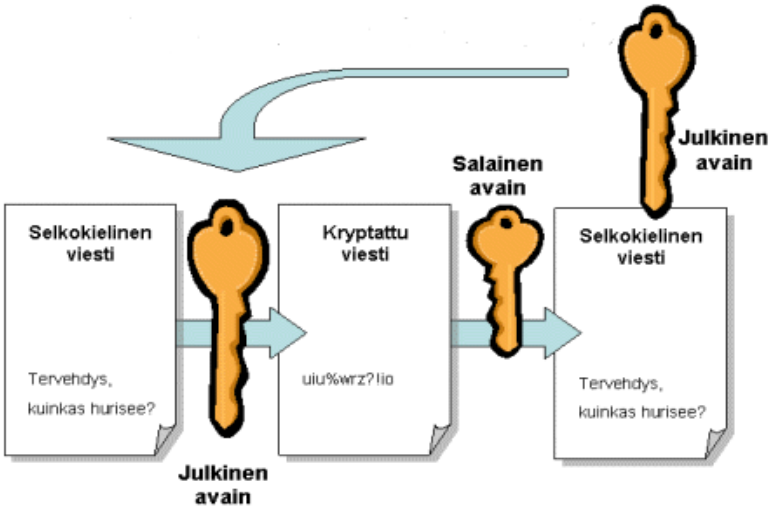


Kuva 1. Symmetrinen salaus [14].

Epäsymmetrinen salausmenetelmä

Epäsymmetrisessä salausmenetelmässä käytetään kahta erillistä avainta, yksityistä avainta ja julkista avainta, jotka yhdessä muodostavat avainparin. Avainparit luodaan samaan aikaan tähän tarkoitetulla ohjelmalla, sillä ne ovat sidoksissa toisiinsa salausmenetelmästä riippuvalla tavalla. Julkinen avain voidaan jakaa kaikille, joiden on tarpeen lähettää tietoa yksityisen avaimen hallitsijalle. Kuka tahansa, jolla on tiedossa julkinen avain, voi lähettää sillä salatun viestin yksityisen avaimen haltijalle käyttäen julkista avainta salaukseen. Salattu viesti voidaan purkaa ainoastaan saman avainparin yksityisellä avaimella. [13.]

Menetelmän etuina ovat julkiset avaimet, joilla osapuolet voivat salata tietoja, jos he tietävät toistensa julkiset avaimet. Näin ollen yksityisiä avaimia ei tarvitse vaihtaa osapuolten välillä kuten symmetrisessä salauksessa, mikä lisää turvallisuutta merkittävästi. Haittapuolina menetelmä on hitaampi kuin symmetrisessä salauksessa, ja niiden toteutukset ovat monimutkaisempia. [13.] Kuva 2 havainnollistaa epäsymmetrisen salausmenetelmän toimintaa.



Kuva 2. Epäsymmetrinen salaus [14].

Useat salausjärjestelmät käyttävät salauksessaan sekä symmetrisiä että epäsymmetrisiä menetelmiä. Näissä hybridisalauksiksi kutsutuissa menetelmissä symmetristä menetelmää käytetään salaamaan symmetrisessä menetelmässä käytetty avain. Tämä toteutetaan niin, että toinen viestinnän kahdesta osapuolesta luo aluksi symmetrisen avaimen, jonka se salaa toisen osapuolen julkisella avaimella ja lähettää sen hänelle. Toimenpiteen jälkeen osapuolet voivat käyttää kyseistä symmetristä avainta viestien salaamiseen. Tällä menetelmällä hyödynnetään molempien menetelmien vahvuuksia. [13.]

Tiedon eheyden varmentaminen

Epäsymmetrisessä salauksessa on myös mahdollista salata viesti käyttäen yksityistä avainta ja avata se julkisella avaimella. Tätä menetelmää käytetään tapauksissa, joissa pyritään todentamaan, onko viesti pysynyt muuttumattomana ja onko se tullut luotetusta lähteestä. Jotta voidaan varmistaa, että tieto on kulkenut muuttumattomana, tarvitaan tiivistefunktio, joka laskee viestistä tiivisteen. Tiivisteen laskemisen yhteydessä käytetään lisäksi yksityistä avainta, joka on vain lähettäjän ja vastaanottajan tiedossa. [13.]

Tiiviste toimitetaan viestin mukana, ja viestin saatuaan vastaanottaja purkaa tiivisteen salauksen käyttäen julkista avainta ja laskee viestistä (pois lukien mukana tulleen tiivisteen) tiivistefunktiota käyttäen tiivisteen ja tämä jälkeen vertaa sitä mukana tulleeseen

tiivisteeseen. Jos tiivisteet täsmäävät täysin, on viestin sisältämä tieto kulkeutunut muuttumattomana ja viestin on allekirjoittanut avainparin yksityisen avaimen hallitseva taho. Jos kuitenkin julkisen avaimen tietää useampi taho, ei tällöin voida olla varmoja tiedon lähettäjästä. Tähän avuksi tulee sähköinen allekirjoitus. [13.]

Jotta salatun viestin vastaanottaja voi olla varma siitä, että salaukseen käytetty julkinen avain kuuluu viestin lähettäjälle, on julkisen avaimen muodostama kokonaisuus yleensä allekirjoitettava sähköisesti. Sähköisen allekirjoituksen suorittaa kolmas osapuoli, johon viestin saaja ja vastaanottaja luottavat. Tallaista osapuolta kutsutaan varmentajaksi (engl. Certificate Authority, CA) ja allekirjoitettua identiteettiä sekä julkisen avaimen muodostamaa kokonaisuutta varmenteeksi. Vastaanottaja voi varmistaa vastaanotetun viestin varmentajalta saadun varmenteen avulla, minkä jälkeen voidaan olla varmoja, että avain kuuluu lähettäjälle. [13.]

Epäsymmetrisiä salaustekniikoita on useita erilaisia. Esittelen seuraavaksi kaksi käytetyintä tekniikkaa, RSA ja ECC. Näistä ECC eli elliptisten käyrien salaus liittyy oleellisesti insinöörityönä toteuttamani prototyypin todentamiseen ja tarkemmin sähköiseen allekirjoittamiseen.

RSA-salausalgoritmi

Tunnetuin ja käytetyin julkisen avaimen salausalgoritmi on RSA (Rivest, Shamir, Adleman), jonka nimi muodostuu kehittäjiensä sukunimien ensimmäisistä kirjaimista. RSA:n turvallisuus perustuu yksisuuntaiseen operaatioon, jossa kaksi suurta alkulukua kerrotaan yhteen. Alkulukujen tulon tekijöihinsä jako on erittäin haastavaa ja aikaa vievää mutta ei mahdotonta. RSA on edelleen laajassa käytössä, mutta salausavaimen kokoa on jouduttu pidentämään merkittävästi sitä mukaa, kuin laskentateho on kasvanut. Pidempien avaimien käyttö aiheuttaa sen, että salatun viestin koko voi olla merkittävästi suurempi kuin alkuperäisen viestin. Tämän vuoksi on alettu siirtyä elliptisten käyrien salausmenetelmiin. [15; 16.]

ECC (Elliptic Curve Cryptography)

Elliptisen käyrän salaus on termi, jota käytetään kuvaamaan salausvälineitä ja protokollia, joiden turvallisuus perustuu erityisiin versioihin diskreettien logaritmien ongelmista.

ECC perustuu numeroihin, jotka ovat liitettyinä matemaattisiin objekteihin, joita kutsutaan elliptisiksi käyriksi. Salaus vastaa epäsymmetristä menetelmää, jossa modulaarisen aritmetiikan sijasta käytetään operaatioita, jotka määritellään elliptisillä käyrillä. [16.]

Elliptisten käyrien käyttö salauksissa on lisääntynyt aiemmasta, sillä se tarjoaa vahvemman turvallisuuden, vähemmän resursseja vaativan algoritmin sekä lyhyemmät avaimien pituudet kuin muut salausalgoritmit. Vertailuna 256 bittiä pitkä ECC-avain tarjoaa saman turvantason kuin 3072 bittiä pitkä RSA-avain. Lyhyemmät avaimet vaativat vähemmän laskentatehoa ja vievät vähemmän kaistanleveyttä tiedonsiirrossa, joten ne ovat varteenotettava tapa salata tietoa IoT-laitteissa. [17.]

3 Turvallinen tiedonsiirto pilvipalveluun

3.1 Tehtävänanto ja aiheen rajaus

Asiakasyrityksellä oli tarve muodostaa MQTT-protokollaa hyödyntävä demoversio IoT-laitteesta, joka lähettää dataa Googlen IoT Core -pilvipalveluun. Erityisesti työssä tuli keskittyä tietoturvaan ja matalataajuuksiseen Narrowband IoT -verkkoteknologiaan. Työn toteutusta varten saatiin asiakasyritykseltä tarvittavat kehitysalustat. Työ on osa suurempaa kokonaisuutta, jonka pohjalta asiakasyritys kehittää omaa IoT-laitetta, jota tarjotaan asiakkaille erilaisiin käyttötarkoituksiin.

Insinöörityön puitteissa tavoitteena oli saada vaadittava teknologia keskustelemaan keskenään, varmistaa turvallinen tiedonsiirto, sekä arvioida Narrowband-verkon toimivuutta datan siirrossa pilvipalveluun. Lopputuotteena oleva prototyyppi on karkea malli valitun teknologian toiminnasta. Kehitysmahdollisuuksia on paljon, mutta ne rajattiin tämän työn ulkopuolelle.

3.2 Narrowband IoT- ja LTE Cat M1 -verkkoteknologiat

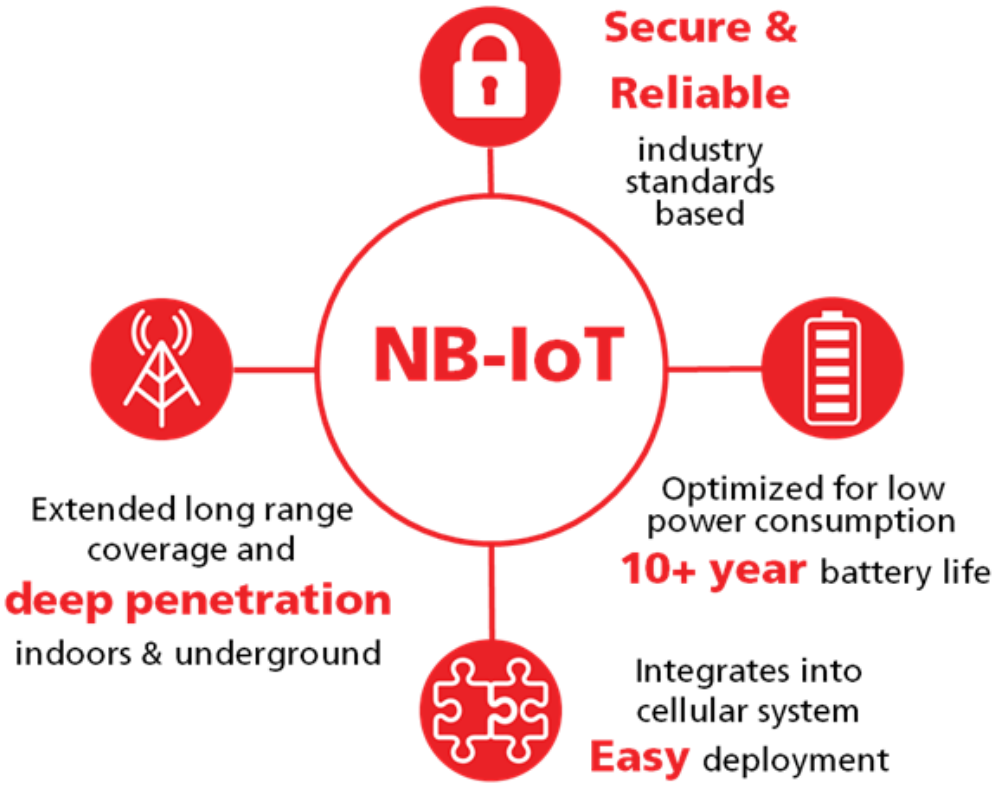
Narrowband IoT (NB-IoT tunnetaan myös nimellä LTE Cat NB1) on uusi langaton teknologia, joka standardoitiin 3GPP:n toimesta vuonna 2016. NB-IoT käyttää vain osan käytettävässä olevasta taajuudestaan, mikä puolestaan aiheuttaa minimaalisen virrankulutuksen. Täten Narrowband-verkkoa käyttävän laitteen käyttöaika paristolla voi olla jopa

yli 10 vuotta. NB-IoT-tekniikka voidaan ottaa käyttöön kaistalla, jossa leveys on vain 200 kHz normaalista noin 10—20 MHz:n kanavasta. Yhden 200 kHz:n kaistan arvioidaan kykenevän tukemaan yli 200 000 NB-IoT-yhteyttä tukiasemaa kohti. Etuna verrattuna muihin IoT-laitteille suunniteltuihin verkkoihin on nopea käyttöönotto jo olemassa olevassa 4G (LTE) -verkossa, joka saadaan aktivoitua asentamalla uusi ohjelmisto tukiasemiin. Näin ollen operaattorien ei tarvitse investoida uusiin laitteisiin ja tukiasemiin. [18; 19.]

Vain noin 200 kHz:n kaistanleveys aiheuttaa hitaan tiedonsiirron verrattaessa muihin langattomiin tiedonsiirtoihin, kuten Wi-Fiin, Bluetoothiin ja matkapuhelinverkkoihin. Kyseinen kaista kykenee lähettämään dataa 27,2 kilobittiä sekunnissa ja lataamaan 62,5 kilobittiä sekunnissa. Tämä nopeus on riittävä IoT-sovelluksille, joille tiedonsiirron nopeudella ei ole suurta merkitystä. Koska NB-IoT:n linkkibudjetti on keskittynyt kapealle taajuuskaistalle, se kykenee saavuttamaan pitkän kantaman pienellä virrankulutuksella.

Testeissä on mitattu noin 5 kilometrin kantamaa kaupunkiolosuhteissa ja jopa 50 kilometrin kantama esteettömässä ympäristössä. Lisäksi kapeakaistaisen taajuuden ansiosta NB-IoT-teknologialla on erinomainen läpäisemiskyky talon rakenteiden lävitse. Tämä on hyvä asia, jos esimerkiksi kellarissa sijaitsevan vesimittarin keräämää dataa halutaan lähettää pilvipalveluun. NB-IoT käyttää lisensoitua taajuutta, joten sillä on parempi signaalin eheys ja vähemmän häiriöitä kuin muilla protokollilla, jotka toimivat samalla taajuudella, kuten Bluetooth ja Wi-Fi. NB-IoT:n vaatima tekniikka on vähemmän monimutkainen kuin perinteisten matkapuhelinverkkojen, mikä yksinkertaistaa OEM-laitteiden suunnittelua, kehittämistä ja käyttöönottoa. Samalla se tarjoaa samat turvallisuusominaisuudet kuin perinteiset LTE-verkot, kuten käyttäjätunnuksen luottamuksellisuuden, tietojen eheyden ja mobiililaitteen tunnistuksen. [20; 21.]

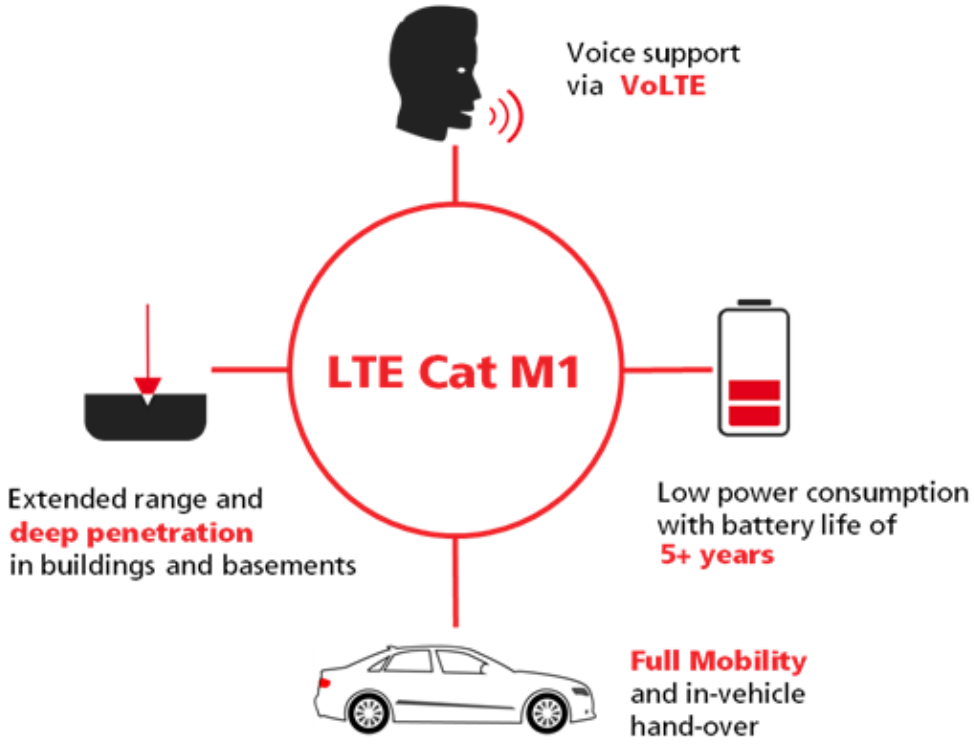
Kirjoitushetkellä NB-IoT-yhteyksiä Suomessa tarjoavat Telia, DNA ja Elisa. Operaattoreista Telia on ottanut teknologian käyttöönsä Suomessa koko verkossaan. Kuva 3 havainnollistaa NB-IoT-verkon hyötyjä.



Kuva 3. NB-IoT-verkko [22].

Samaan aikaan NB-IoT:n kanssa standardoitiin 3GPP:n toimesta myös langaton teknologia LTE Cat M1, joka on kehitetty IoT-laitteita varten. LTE Cat M1:n etuina ovat suurempi datasiirron lataus ja lähetysnopeudet, jotka ovat 375 kilobittiä sekunnissa. Näillä nopeuksilla LTE Cat M1 voi toimittaa FOTA (firmware over the air) -päivityspaketteja laitteille kohtuullisin aikavälein, joten se soveltuu hyvin kriittisiin sovelluksiin, joita käytetään pitkiä aikoja.

Verrattuna NB-IoT-teknologiaan LTE Cat M1 osaa siirtää meneillään olevan tiedonsiirtoprosessin kahden tukiaseman välillä. Tästä havainnollinen esimerkki on kylmäkuljetuksen lämpötilan mittaus. Vaikka kuljetus siirtyy matkan aikana tukiaseman piiristä toiseen, toimii IoT-laitteen yhteys kuten matkapuhelimen, eikä se menetä yhteyttään siirryttäessä kahden tukiaseman välillä. Vastaavassa tilanteessa NB-IoT-teknologiaa käyttävän laitteen olisi luotava yhteys uudelleen tukiaseman peittoalueen vaihtuessa, joten olemassa oleva yhteys katkeaisi. Toinen etu on ääniominaisuuden tuki VoLTE:n kautta (voice over LTE), jolla voidaan luoda IoT-sovelluksia esimerkiksi tiettyihin terveys- ja turvallisuusohjelmiin (esim. stay-in-place-ratkaisut ja hälytyspaneelit). [23.] Kuva 4 havainnollistaa LTE Cat M1 -verkon toiminnallisuuksia.



Kuva 4. LTE Cat M1 -verkko [23].

Vaikka NB-IoT- ja LTE Cat M1 -teknologiat ovat hyvin samankaltaisia niille on olemassa omat käyttökohteensa. LTE Cat M1:n tarjoama nopeampi tiedonsiirto ja automaattinen kättely tukiaseman vaihdon välillä on hyvä ratkaisu, jos IoT-laitetta liikutellaan paljon esimerkiksi ajoneuvojen jatkuvaan paikantamiseen tai liikuteltavaa dataa on paljon. NB-IoT-teknologia sopii vähän tiedonsiirtokapasiteettia vaativiin sovelluksiin, joiden olinpaikka pysyy muuttumattomana ja jotka voivat sijaita heikosti signaalia lävitse päästävien esteiden takana tai esimerkiksi maan alla, kuten kellarissa tai viemäristössä.

3.3 Google Cloud IoT Core ja sen vaatimat tekniikat

IoT-laiteella yleensä kerätään dataa, jonka vastaanottamiseen tarvitaan datankerääjä. Datankerääjiä on tarjolla useiden eri yritysten toimesta mutta asiakasyrityksen pyynnöstä insinöörityössä käytettiin Googlen tarjoamaa datankerääjää. Google IoT Core tarjoaa kokonaisvaltaisen ratkaisun IoT-laitteiden lähettämien tietojen keräämiseen, käsittelyyn, analysointiin ja visualisointiin. Se tukee yleisiä MQTT (Message Queue Telemetry Transport)- ja HTTP-tiedonsiirtoprotokollia. Projektissa oli tarkoituksena hyödyntää MQTT-protokollaa, joten esittelen sen tässä lyhyesti.

3.3.1 MQTT-protokolla

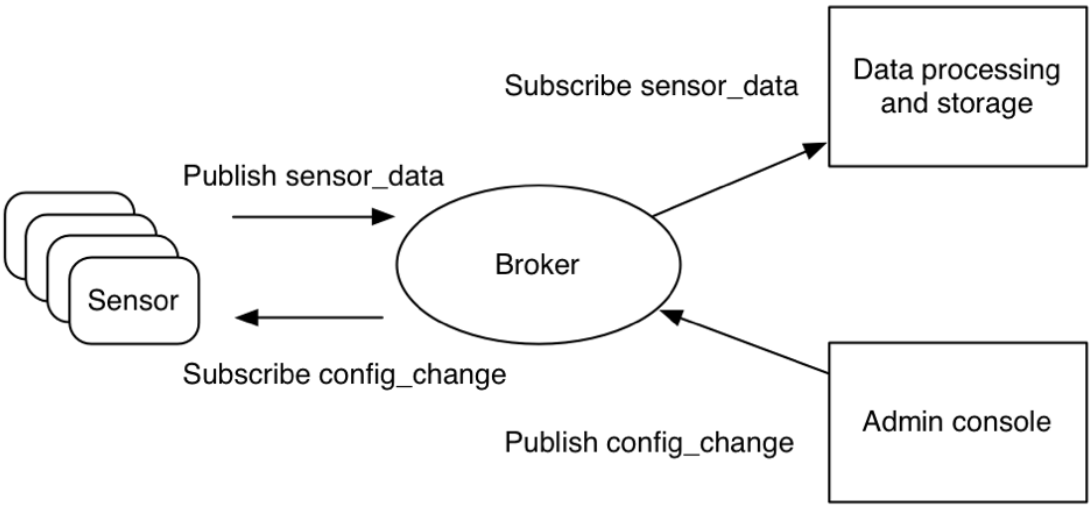
MQTT:n ovat kehittäneet IBM:n Andy Stanford-Clark ja Arcomin Arlen Nipper vuonna 1999. MQTT on yksinkertainen ja erittäin kevyt tiedonsiirtoprotokolla, joka toimii TCP/IP-protokollan päällä, ja siitä on tullut standardi IoT-laitteiden viestinnälle. Se on suunniteltu vähän virtaa ja tiedonsiirtokapasiteettia käyttäville laitteille, samalla kuitenkin säilyttäen luotettavan ja toimintavarman yhteyden. Protokolla on ideaalinen esineiden internet -laitteille, joissa kaistanleveys ja akun kapasiteetti ovat rajoitettuja. Nimestään huolimatta MQTT-protokollalla ei ole mitään tekemistä viestijonojen kanssa, vaan se käyttää hyväkseen julkaisu- ja tilausmallia (publish/subscribe). [24; 25.]

MQTT-istunto on jaettu neljään vaiheeseen: yhteyden muodostaminen, todennus, viestintä ja viestinnän päättyminen. MQTT:tä varten on varattu TCP/IP-portti 1883 salaamatonta tiedonsiirtoa varten ja salattua tiedonsiirtoa varten portti 8883, joka käyttää SSL- ja TLS-salausprotokollia. [26.]

Julkaise tai tilaa -viestintämalli

MQTT koostuu kahdesta erillisestä yksiköstä, viestin välittäjästä (broker) ja asiakkaista (client). Viestin välittäjänä toimii palvelin, joka vastaanottaa viestit asiakkailta ja reitittää ne kohdeasiakkaille. Asiakas voi olla kuka tahansa, joka voi olla yhteydessä välittäjään, esimerkiksi IoT-anturi tai pilvessä toimiva sovellus, joka käsittelee IoT-dataa. Kun asiakas ottaa yhteyden, yhteys voi olla joko tavallinen salaamaton TCP/IP-yhteys tai salattu TLS-yhteys, jos viestin sisältö on arkaluonteista. Asiakas julkaisee viestin lähettämällä viestin ja aiheen (topic) välittäjälle. Välittäjä tämä jälkeen välittää viestin kaikille asiakkaille, jotka ovat tilanneet kyseisen aiheen. [25.]

MQTT-viestit jaotellaan eri aiheisiin, jolloin kehittäjien on mahdollista määrittää tietyn asiakkaan olevan ainoastaan tekemisissä tiettyjen viestien kanssa. Esimerkiksi kuten kuvassa 5 esitetään, anturi julkaisee arvon ”sensor_data”-aiheella ja tilaa välittäjältä ”config_change”-aiheen. Tietojenkäsittelysovellus, joka tallentaa anturilta tulevan tiedon tietokantaan, tilaa välittäjältä ”sensor_data”-aiheen. Verkon ylläpitäjä voi säätää anturin konfiguraatiot, kuten herkkyyden ja näytteenottotaajuuden, ja julkaista nämä muutokset ”config_change”-aiheeseen. [25.]



Kuva 5. MQTT-protokollan toiminta [25].

MQTT-protokollaa käyttävät sovellukset voivat käyttää tietosisältönä (payload) useaa eri dataformaattia, kuten JSON, XML, salattu binääri tai base64, joita se voi julkaista välittäjälle, kunhan kohdeasiakkaat ovat kykeneviä käsittelemään kyseistä formaattia. [25.]

Asiakkaan todentaminen ei ole yksinkertainen prosessi. MQTT ei tarjoa työkaluja sen hallinnointiin, ketkä omistavat aiheen ja ketkä voivat lähettää dataa kyseiseen aiheeseen. Tämän ansiosta verkossa on helppo lähettää välittäjälle vahingollisia viestejä tahallaan tai virheellisesti. Lisäksi viestin vastaanottajalla ei ole mahdollisuutta tietää, kuka lähetti alkuperäisen viestin, ellei tietoja ole varsinaisessa viestissä. Koska MQTT-protokolla on suunniteltu resurssirajoitteisille laitteille, ei SSL/TLS-yhteyden toteuttaminen ole välttämättä mahdollista tai se ei ole toivottavaa. Tällaisissa tapauksissa käyttäjän todentaminen on toteutettu käyttäjänimen ja salasanan lähettämisessä yhteyden muodostamisen yhteydessä. [26.]

Asiakkaan todentamiseen datankerääjä hyödyntää JSON Web Token -menetelmää, josta kerrotaan seuraavassa luvussa.

3.3.2 JSON Web Token -menetelmä

JSON Web Token (JWT) on avoin standardi, joka määrittää kompaktin ja itsenäisen tavan lähettää tietoa turvallisesti kahden osapuolen välillä. Yleisimpiä JWT:n käyttötarkoituksia on käyttäjän todentaminen ja tietojen vaihtaminen. JWT:n sisältämä tieto voidaan todentaa ja siihen voidaan luottaa, sillä se on digitaalisesti allekirjoitettu. Täten voidaan

olla varmoja, että tiedon lähettäjät ovat niitä, joita he väittävät olevansa. JSON Web Tokenit allekirjoitetaan käyttäen yhteistä salaisuutta tai julkisen avaimen salausmenetelmillä, kuten RSA:lla käyttäen yksityistä avainta. Pienen kokonsa ansiosta JWT voidaan lähettää URL-osoitteen kautta POST-parametrin sisällä tai http-otsikon sisällä. [27.]

JSON Web Token koostuu kolmesta osasta, jotka ovat eroteltuina toisistaan pisteillä. Osat ovat otsikko (header), hyötykuorma (payload) ja allekirjoitus (signature), jotka kaikki ovat base64Url-muotoon koodattuja.

Otsikko koostuu tyypillisesti kahdesta eri osasta: merkkijonon (token) tyypistä, joka on JWT, ja salausalgoritmin tyypistä, jota allekirjoitukseen on käytetty. Kuva 6 havainnollistaa otsikon sisältämää dataa, jossa allekirjoitusalgoritmiksi on määritetty ES256-algoritmi (ECDSA-allekirjoitus + SHA-256-tiivistealgoritmi) ja merkkijonon tyypiksi JWT.

```
{
  "alg": "ES256",
  "typ": "JWT"
}
```

Kuva 6. JSON Web Token -otsikko [28].

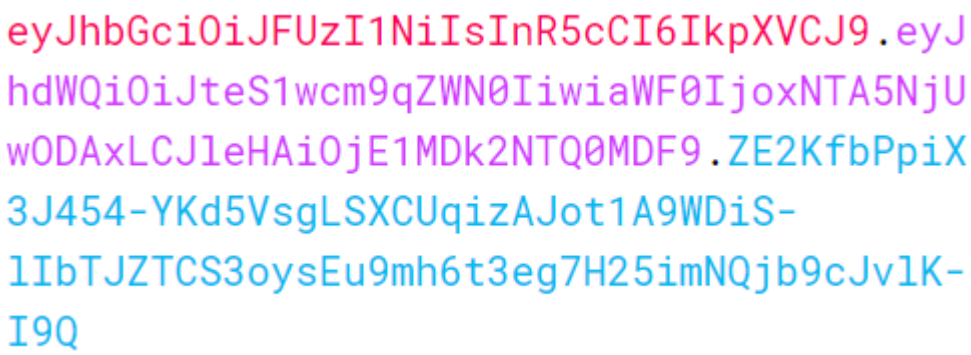
Hyötykuorma sisältää väitteet (claims), jota voidaan tyypillisesti käyttää käyttäjien tunnistetietojen ja identiteetin välittämiseen tai muuhun informaatioon, jota prosessi vaatii. Kuvassa 7, aud (audience) kuvaa, kenelle tieto on tarkoitettu, iat (issued at) kuvaa aikaleimaa, jolloin JWT on luotu, ja exp (expiration) aikaa, jolloin JWT:n voimassaolo lakkaa.

```
{
  "aud": "my-project",
  "iat": 1509650801,
  "exp": 1509654401
}
```

Kuva 7. JSON Web Token -hyötykuorma [28].

Allekirjoitus koostuu base64Url-koodatuista otsikosta ja hyötykuormasta. Base64Url-koodatut otsikko ja hyötykuorma yhdistetään, minkä jälkeen se allekirjoitetaan käyttäen otsikossa määritettyä algoritmia. Myös allekirjoitus base64Url koodataan ja lisätään base64Url-koodatun otsikon ja hyötykuorman perään pisteellä erotettuna, jolloin tuloksena on JWT.

Kuva 8 näyttää valmiin JSON Web Tokenin, jossa on käytetty aikaisemmissa kuvissa olevaa otsikkoa (punainen merkkijono) ja hyötykuormaa (violetti merkkijono). Sinisellä värillä oleva merkkijono on allekirjoitus otsikosta ja hyötykuormasta.



Kuva 8. Valmis JSON Web Token. [28.]

JSON Web Tokenin vastaanottavalla osapuolella on oltava käytössään allekirjoitukseen käytetyn yksityisen avaimen julkinen avain tai yhteinen salaisuus, jolla vastaanottaja voi purkaa allekirjoituksen, jolloin tuloksena on tiiviste viestin base64Url-koodatuista otsikosta ja hyötykuormasta. Vastaanottaja laskee myös itse tiivisteen viestin otsikosta ja hyötykuormasta. Jos laskettu tiiviste ja julkisella avaimella tai yhteisellä salaisuudella purettu allekirjoituksen tiiviste ovat täysin samat, voidaan todeta, että tieto on tullut luotetusta lähteestä ja tämä tieto on pysynyt siirron aikana muuttumattomana, sillä pienikin muutos viestissä tekee tiivisteestä erilaisen.

Base64 ja base64Url

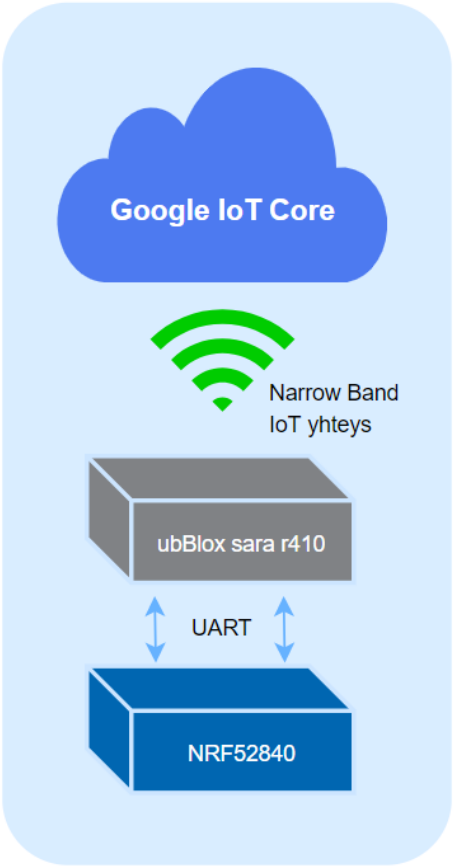
Base64-koodausta käytetään yleensä silloin, kun on tarpeellista lähettää binääritietoa sellaisten laitteiden kautta, jotka on kehitetty käsittelemään tekstimuodossa olevaa informaatiota. Base64-koodaus käyttää ainoastaan kirjaimia, numeroita ja merkkejä + ja /, millä pyritään takaaman se, että näitä merkkejä ymmärtävät kaikki osapuolet ja että tieto pysyy muuttumattomana siirron aikana erilaisten laitteiden välillä.

Base64-koodaus itsessään ei toimi kovin hyvin URL-osoitteissa, sillä merkit +, / ja =, joita käytetään base64-koodaamisessa, ovat varattuja tai niillä on erityinen merkitys URL-osoitteissa. Tätä ongelmaa varten on kehitetty base64Url-koodaus, joka on muuten sama kuin base64-koodaus mutta merkit + ja / on korvattu merkeillä - ja _. [29; 30.]

4 IoT-oletusyhdyskäytävän toteutus

4.1 Käytetyt laitteet ja teknologiat

Narrowband IoT -yhteyden luomiseen sain asiakasyritykseltä käyttööni u-blox SARA-R410M-02B-kehitysalustan, joka toimii modeemina. Lisäksi sain Nordic Semiconductorin NRF52840-kehitysalustan, jolla komennetaan modeemia AT-komennoilla ja jolla luodaan tarvittava MQTT-tiedonsiirtoprotokolla. Kuvassa 9 havainnollistetaan prototyypin toimintaa.



Kuva 9. Havainnekuva prototyypin kokonaisuudesta.

U-Blox sara-R410M-02B -kehitysalusta

SARA-R410-02B-kehitysalusta sisältää LTE Cat M1- ja NB1-moduulit, jotka toimivat globaalisti ympäri maailmaa. Modeemi tukee muun muassa http-, HTTPS- ja TLS-protokollia ja siinä on TCP/UDP-pino ja FOTA (firmware over the air)-tuki. Modeemin toimintaympäristön lämpötilan tulee olla -40:n ja 85 celsiusasteen välillä. Modeemi tarjoaa erittäin alhaisen virrankulutuksen, joten SARA-R410M-02B sopii IoT-laitteiden kommunikointiväyläksi. [31.] Kuvassa 10 on esiteltynä projektissa käytetty SARA-R410-02B-kehitysalusta.

Kuva 10. U-blox SARA-R410M-02B -kehitysalusta [31].

Nordic Semiconductor nRF52840 -kehitysalusta

NRF52840 on erittäin vähävirtainen ARM Cortex-M4 -suorittimen ympärille suunniteltu 2,4 GHz:n taajuutta käyttävä järjestelmäpiiri. Siinä on 1 Mt flash-välimuistia ja 256 kilobittiä RAM-muistia, ja se tukee monia 2,4 GHz:n taajuutta käyttäviä protokollia, kuten Bluetooth, Bluetooth low energy (BLE) ja ANT. NRF5240:n tuettuina kehitystyökaluina ovat Segger Embedded Studio, keil IAR ja GCC. [32.] Kuvassa 11 on esiteltynä projektissa käytetty nRF52840-kehitysalusta.

Kuva 11. NRF52840-mikrokontrolleri [32].

NRF52840-kehitysalusta valikoitiin projektiin, koska tämä piiri tulee olemaan asiakasyrityksen valmiissa tuotteessa.

Nordic Semiconductorin NRF52840-mikrokontrollerin kanssa käytin uVision Keil 5 -ohjelmointiympäristöä ja Nordic Semiconductorin nRF5-kehitysympäristöä, joka tarjoaa tarvittavat laiteajurit ja runsaasti valmiita esimerkkikoodeja.

4.2 Tiedonsiirron tekninen toteutus

Kommunikointi modeemin ja mikrokontrollerin välillä tapahtui UART (Universal Asynchronous Receiver Transmitter) -tietoliikennemuotoa käyttämällä.

UART käyttää tiedonsiirrossa kolmea signaalia: tx (lähetetty sarjadata), rx (vastaanotettu sarjadata) ja maa. UART lähettää dataa asynkronisesti, mikä tarkoittaa, että tiedonsiirron nopeutta ei kerrota tiedon vastaanottajalle erillisellä signaalilla. Sen sijaan lähettävä UART lisää aloitus- ja lopetusbitit siirrettävään datapakettiin, ja ne määrittävät datapaketin alun ja lopun niin, että vastaanottava UART tietää, milloin se voi alkaa lukea bittejä. Kun vastaanottava UART havaitsee aloitusbitin, se alkaa lukea saapuvia bittejä tietyllä taajuudella, jota kutsutaan baudrateksi (siirtonopeus). Baudrate on datan siirron mitta, joka ilmaistaan bitteinä sekunneissa. [33.] Toimiakseen lähettäjän ja vastaanottajan UART täytyi määrittää samalle siirtonopeudelle. Lisäksi viestin virtauksen hallinnoimiseen käytettiin RTS (ready to send)- ja CTS (clear to send) -signaaleja.

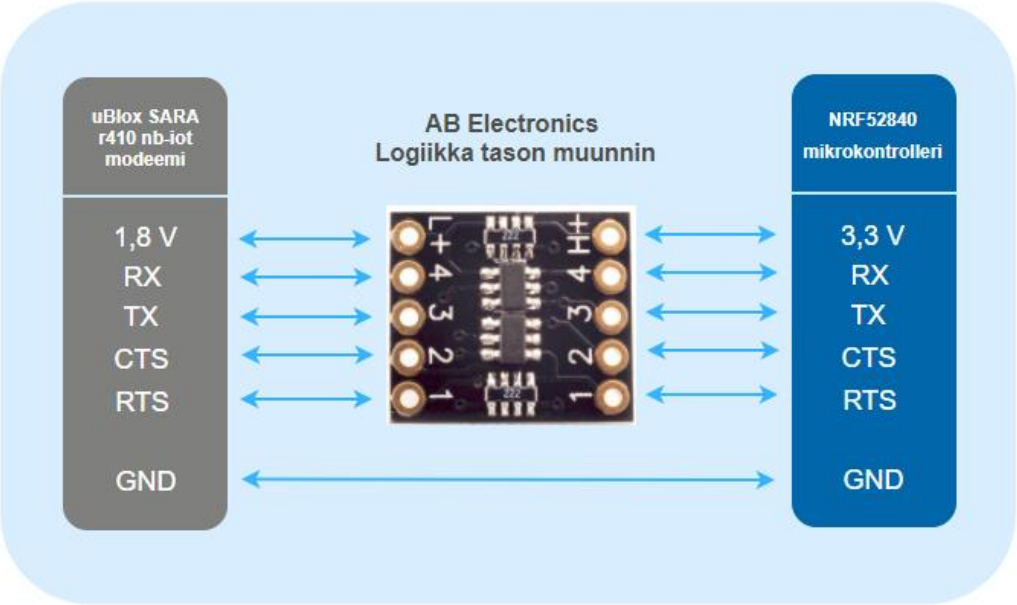
NRF52-mikrokontrollerin UART käyttää 3,3 voltin jännitettä, kun taas u-bloxin modeemi käyttää 1,8 voltin jännitettä. Suuren jännitteen vuoksi riskinä oli modeemin rikkoutuminen, joten laitteiden väliin tarvittiin kaksisuuntaisen logiikkatason muuntaja. Näin pystyttiin takaamaan laadukas ja turvallinen kommunikointi laitteiden välillä. Yksinkertaisesti

selitettynä logiikkatason muuntaja muuntaa korkean tason jännitteen matalan tason jännitteeksi ja toisin päin. Tässä tapauksessa nRF52-kehitysalustalta tuleva lähetyssignaali muutetaan 3,3 voltista 1,8 volttiin ja modeemin lähetyssignaali 1,8 voltista 3,3 volttiin.

Projektissa käytettiin Metropolia Ammattikorkeakoululta saatua AB Electronicsin valmistamaa logiikkatason muunninta.

SARA-R4-modeemissa sijaitsee kuusi kytkintä, joilla voidaan ohjata joitain kehitysalustan toimintoja, kuten se, mitä kommunikaatio väylää käytetään. Mahdollisia kommunikaatioväyliä ovat MINIUSB, RS232 tai UART jonka DIL B2B-pinnejä käyttämällä saadaan suora yhteys modeemille. Valmiissa tuotteessa mikrokontrolleri tulee olemaan suoraan kytkettynä modeemiin, joten prototyypissä käytettiin samaa kytkentää. Näin ollen yhdistettiin mikrokontrollerin rx-, tx-, cts-, rts- ja ground-pinnit modeemin DIL B2B vastaaviin pinneihin ja asetettiin kytkimet käyttämään kommunikaatioväyläksi DIL B2B -yhteyttä. Tämä tapahtui asettamalla kytkin SW401 asentoon B2B.

Kuva 12 havainnollistaa UART-yhteyttä mikrokontrollerin ja modeemin kytkentää logiikkatason muuntajan lävitse.



Kuva 12. Laitteiden kommunikointi logiikkatason muuntajan kautta.

Nordic Semiconductorin SDK:n mukana toimitetuista esimerkeistä löytyi UART-esimerkkikoodi, jonka ympärille lähdettiin rakentamaan toteutusta. U-bloxin modeemiin oli määritelty seuraavat asetukset UART-kommunikaatiolle:

- datanopeus: 115,200 bittiä sekunnissa
- databitit: 8
- pariteetti: N
- lopetusbitit: 1
- virtauskontrolli: laitteisto.

Samat asetukset määriteltiin myös mikrokontrollerin UART-asetuksiin. Tämän jälkeen voitiin kokeilla yhteyden toimivuutta. Modeemia ohjattiin käyttäen AT-komentoja, joita lähetettiin UART:n kautta mikrokontrollerilta modeemille. Yksinkertaisimpina komentoina modeemille voitiin lähettää komento OK, johon modeemin pitäisi vastata takaisin OK. Tällä komennolla voitiin varmistaa kommunikaatioyhteyden toiminen. Ensimmäisillä kytkennöillä modeemi ei vastannut mitään takaisin.

Syynä tähän oli väärä kytketä laitteiden välillä. Tavallisesti tx- ja rx-signaalit ristiinkytketään niin, että tiedon lähettävän laitteen tx (datan lähetys)-signaali kytketään vastaanottavan laitteen rx (datan vastaanotto)-porttiin. Logiikkatason muunnin kuitenkin teki tämän itse, joten signaalit oli yhdistettävä suoraan rx-portista rx-porttiin ja tx-portista tx-porttiin. Tämän jälkeen yhteys toimi ja modeemilta saatiin OK-vastaus takaisin.

Saatuani modeemin ja mikrokontrollerin välisen kommunikoinnin toimimaan, kokeilin modeemin internetyhteyttä. SIM-kortin sain käyttööni asiakasyritykseltä, ja se asennettiin modeemin SIM-korttipaikkaan. SIM-korttiin ei ollut asetettu PIN-koodia, joten sen avaaminen ei tässä tapauksessa vaatinut erillisiä AT-komentoja. Modeemi osaa itse yhdistää itsensä verkkoon, mutta se ei tässä aina välttämättä onnistu tai verkkoa ei ole saatavilla, joten verkon tila joudutaan tarkistamaan AT-komennoilla.

Modeemin käyttöohjeen mukana tuli esimerkki, miten luodaan TCP-yhteys, jonka avulla voidaan varmistaa yhteys. Jos AT-komento on tunnistettu ja se on suoritettavissa, modeemi palauttaa vastauksena OK. Muussa tapauksessa modeemi palauttaa virhekoodin. Modeemi vaatii jokaisen lähetetyn AT-komennon perään \r (carriage return) -merkin, joka kuvaa komennon päättymistä. Taulukko 1 havainnollistaa yhteyden luomiseen käytettyjä AT-komentoja.

Taulukko 1. TCP-yhteyden luominen.

AT komento	Vastaus modeemilta	Kuvaus
AT+CGATT?	+CGATT: 1 OK	Varmistetaan, onko moduuli kytkettynä verkkoon.
AT+CEREG?	+CEREG: 0,1 OK	Varmistetaan verkon rekisteröinnin tila.
AT+USOCR=6	+USOCR: 0 OK	Luodaan TCP-kanta.
AT+USOCO=0, "195.34.89.241", 7	OK +UUSORD: 0,32	Yhdistetään ubloxin-palvelimelle.
AT+USORD=0, 32	+USORD: 0,32,"u-blox AG TCP/UDP test service" OK	Vastaanotetaan viesti ubloxin-palvelimelta.
AT+USOCL=0	OK	Suljetaan TCP-kanta.

Yhteys toimi ensi yrittämältä, ja modeemi sai vastaanotettua viestin palvelimelta. Yhteyden toimiessa haluttiin lisäksi olla varmoja, että modeemi käyttää ainoastaan NB-IoT-moduulia tiedonsiirtoon, ja lähetettiin taulukossa 2 esitelty AT-komento modeemille.

Taulukko 2. Vahvistetaan NB-IoT-moduulin käyttö.

AT komento	Vastaus modeemilta	Kuvaus
AT+URAT=8	+URAT: 8 OK	Määritetään radioteknologiaksi 8, joka vastaa NB IoT-radioteknologiaa modeemissa.

Yhteyden toimiessa voitiin aloittaa tekemään MQTT-yhteyttä mikrokontrollerilla. MQTT-kirjastoksi valittiin avoimen lähdekoodin Eclipse Paho Embedded MQTTPacket -kirjasto, joka on suunniteltu sulautetuille järjestelmille. Se sisältää alimman tason C-kielen kirjaston MQTT:n pienimmillä vaatimuksilla ilman internetyhteyden käsittelyä. Kirjasto luo ainoastaan tarvittavan MQTT-paketin, joka lähetetään modeemille AT-komentoja käyttämällä.

MQTT-paketin lähetystä lähdettiin kokeilemaan Eclipsen tarjoamalle julkiselle MQTT-välittäjälle. MQTT-paketti luodaan antamalla tarvittavat tiedot välittäjäpalvelimesta. Eclipsen MQTT-välittäjän palvelimen nimi on iot.eclipse.org ja porttina on 1883. Tämä lisäksi tarvitaan aiheen nimi ja tietosisältö, jossa on viesti, joka halutaan lähettää kyseiseen aiheeseen. MQTT-kirjasto tarjoaa myös käyttäjänimi- ja salasanavaihtoehdot käyttäjän todentamiseen välittäjän päässä, mutta ne jätettiin tyhjiksi testiä varten. Valitsin aiheeksi ”/sensodata” ja tietosisältöön loin seuraavanlaisen viestin: ”temperature:22, humidity:42”.

MQTT-paketti luodaan käyttämällä kolmea funktiota, joista jokainen palauttaa viestin koon kokonaislukutyyppiseen muuttujaan. Viestin koon tietäminen on tärkeää, sillä modeemi haluaa ennalta tietää, kuinka merkkiä pitkä lähetettävä data tulee olemaan. Käyttämällä näitä funktioita kirjasto luo niille parametrina syötetylle char-tyyppiseen listaan valmiin MQTT-paketin, joka voidaan lähettää modeemille.

Ennen paketin syöttämistä modeemille modeemissa pitää avata TCP-yhteys Eclipsen MQTT-välittäjäpalvelimelle. Yhteyden luominen tarvitsee palvelimen IP-osoitteen, jonka sain lähettämällä komennon ”ping www.iot.eclipse.org” Windowsin komentorivityökaluun. Yhteyden luominen toteutettiin taulukossa 3 esitetyillä AT-komennoilla.

Taulukko 3. Yhteyden luominen MQTT-välittäjälle.

AT-komento	Vastaus modeemilta	Kuvaus
AT+USOCR=6	+USOCR: 0 OK	Luodaan TCP-kanta.
AT+USOCO=0, "184.41.30.241", 1883	OK AT+USOCO=0, "184.41.30.241", 1883	Yhdistetään Eclipsen MQTT-välittäjäpalvelimelle.
AT+USOWR=0, 67	AT+USOWR=0, 67 @	Valmistellaan modeemi ottamaa vastaan 67 tavua pitkä merkkijono, joka syötetään TCP 0 -kantaan. @-merkki kuvaa modeemin valmiutta ottaa data vastaan.

Yhteys on avattu välittäjälle ja voidaan syöttää MQTT-paketti modeemille. Lähetettyään paketin onnistuneesti modeemi vastaa takaisin komennolla OK.

Jotta pystyttiin olemaan varmoja, että viesti tosiaan saavutti Eclipsen välittäjän, asennettiin Chrome-selaimeen MQTTlens-lisäosa, jolla pystyttiin tilaamaan ennalta määritelty /sensordata-aihe välittäjältä. Kun MQTT-paketti lähetettiin mikrokontrollerilta, MQTTlens-ohjelman tilausosaan ilmestyi mikrokontrollerilta lähetetty viesti. Tällä voitiin todeta, että lähetetty MQTT-paketti päätyi välittäjälle ja aiheen tilattua saatiin kaikki viestit, mitä välittäjälle modeemilta lähetettiin. Tällä testillä todettiin MQTT-kirjaston ja MQTT-paketin lähettäminen ilman salausta toimivaksi.

JSON Web Tokenin luominen nRF52-kehitysalustalla

Seuraavaksi tarkoituksena oli lähettää MQTT-paketteja Googlen IoT Coreen eli datankerääjään. Verrattuna aiemmin toteutettuun MQTT-yhteyteen, datankerääjän MQTT-välittäjä vaatii toimiakseen TLS-suojatun yhteyden, ja MQTT-paketin salasanaksi vaaditaan JSON Web Token, jota datankerääjä käyttää laitteen todentamiseen. Käyttäjänimeä datankerääjän MQTT-välittäjä ei tarvitse, joten sen voi jättää tyhjäksi tai käyttää nimenä ”unused” (käyttämätön).

Datankerääjä tukee RSA:ta ja elliptisten käyrien salausmenetelmää JSON Web Tokenin allekirjoituksen todentamiseen. Allekirjoitusalgoritmiksi valittiin elliptisten käyrien allekirjoitusalgoritmi (ECDSA) P-256-käyrällä ja SHA-256-tiivistealgoritmilla, joka tunnetaan JSON Web Token ympäristössä lyhenteellä ES256. Kuten luvussa 2.4 tiedon salaaminen internetissä todettiin, elliptisten käyrien salausmenetelmät sopivat toiminnallisuuksiensa puolesta parhaiten tähän projektiin.

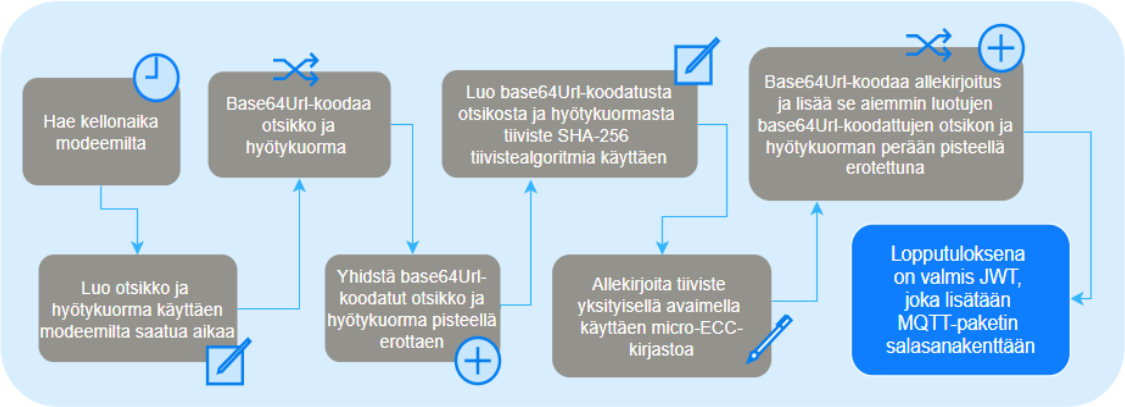
IoT-laitteelle ja datankerääjälle luotiin avainpari OpenSSL-ohjelmalla. Avainparin julkinen avain syötettiin datankerääjään, ja yksityistä avainta käytettiin JSON Web Tokenin allekirjoittamiseen mikrokontrollerissa. Avainparin generoinnin ja julkisen avaimen syöttämisen datankerääjään teki asiakasyrityksen työntekijä.

Google tarjoaa esimerkkikoodin JWT:n luomisesta ja viestin lähettämisestä MQTT-protokollalla C-ohjelmointikielellä. Esimerkkikoodi käyttää apunaan useita kirjastoja, joista kaikkia ei ole suunniteltu käytettäväksi sulautetuilla järjestelmillä. Ne ovat joko liian raskaita tai tarkoitettu käytettäväksi Linux- tai Windows-ympäristöissä, joten JSON Web Token piti luoda kevyemmillä menetelmillä, jotka soveltuvat ARM-suorittimille ja sulautetuille järjestelmille.

JSON Web Tokenin allekirjoitus vaatii kryptografiakirjaston. Käytin Nordic Semiconductorin omaa kryptografiakirjastoa, joka käyttää hyväksi micro-ecc-kirjastoa. Kirjasto on pieni ja nopea, ja sillä voidaan luoda avainpareja, salata tietoja sekä allekirjoittaa dataa. Micro-ecc-kirjaston tuominen suoraan Keil-ohjelmointiympäristöön ja kääntäminen aiheutti kuitenkin virheen käännösvaiheessa. Ongelmaksi paljastui eri syntaksi kuin GCC-kääntäjällä, jolla micro-ecc-kirjasto on tarkoitettu käännettäväksi.

Avuksi tähän löytyi ohje Nordic Semiconductorin cryptography library -dokumentoinnista, jossa neuvotaan kääntämään micro-ecc-kirjasto suoraan kehitysympäristön micro-ecc-kansioon make-komennolla käyttäen GCC-kääntäjää. Tämä jälkeen Keil-ohjelmointiympäristö osasi hakea itse käännetyn kirjaston kyseisestä sijainnista ja sain käyttööni micro-ecc-kirjaston toiminnallisuudet nRF52-kehitysalustalla.

Tarvittavien kirjastojen käyttöönoton jälkeen oli aika toteuttaa JSON Web Token. Kuten luvussa 3.3.2 mainitsin, JSON Web Token koostuu kolmesta osasta, jotka ovat otsikko (header), hyötykuorma (payload) ja allekirjoitus (signature). Kuva 13 havainnollistaa JWT:n toteutuksen projektissa.



Kuva 13. JSON Web Tokenin luomisen kulku insinöörityön prototyypissä.

Otsikko koostuu kahdesta osasta, jotka ovat tyyppi ja algoritmi. Tyypiksi valittiin JWT ja algoritmiksi ES256. Nämä tiedot sisällytetään JSON-muotoon char-tyyppiseen taulukkoon. JWT-otsikko näyttää JSON-muodossa seuraavalta:

{"alg": "ES256", "typ": "JWT"}

JWT:n hyötykuorma, joka sisältää väitteet, vaatii audiencen (yleisö), joka sisältää Google Cloud -projektin ID-osoitteen, jonne laite on rekisteröity, aikaleiman (iat), jolloin JWT on luotu ja JWT:n voimassaoloajan päättymisen aikaleiman (exp). Molemmat aikaleimat pitää esittää UNIX-ajassa, joka ilmaisee ajan sekunteina ajanhetkestä 1.1.1970 kello 00:00:00 lähtien. Aikaleima toteutettiin kysymällä modeemilta aikaa. Kellonajan kysynnälle luotiin oma funktio, joka kysyy AT-komennolla ”AT+CCLK?” kellonaikaa modeemilta. Modeemi palauttaa ajan seuraavanlaisessa muodossa: ”vuosi/ kuukausi/ päivä, tunnit, minuutit, sekunnit”. Jotta modeemi osaa antaa ajan, sen pitää olla rekisteröityneenä verkkoon. Ajan hakeminen modeemilta toteutettiin taulukossa 4 esitetyllä AT-komennolla.

Taulukko 4. Ajan hakeminen modeemilta.

AT-komento	Vastaus modeemilta	Kuvaus
AT+CCLK?	+CCLK: "12/05/23,21:54:21+00"	Palauttaa modeemilta päivän ja kellonajan.

Ajan saamisen jälkeen funktio erottelee päivän, kuukauden, vuoden ja kellonajan omiin muuttujiinsa. Muuttujat lisätään struktuuriin, joka syötetään C-kielen time-kirjaston

mktime-funktioon, joka luo annetusta ajasta UNIX-ajan. Saatu UNIX-aika liitetään hyötykuorman iat (issuden at) parametriksi. Hyötykuormassa oleva exp (expiration) kertoo JWT:n voimassaoloajan päättymisajankohdan. Parametriksi exp:lle annetaan sama UNIX-aikaleima kuin iat:n parametriksi annettiin, mutta exp-aikaleimaan lisättiin lisäksi haluttu voimassaoloaika sekunteina. Voimassaoloajaksi annettiin tunti, joka saadaan lisäämällä UNIX-aikaan luku 3 600 joka viittaa 3 600 sekuntiin. Datankerääjä hyväksyy maksimivoimassaoloajaksi 24 tuntia. Työssä käytetty JWT:n hyötykuorma näyttää JSON-muodossa seuraavalta:

```
{
  "aud":"my-project",
  "iat":1509650801,
  "exp":1509654401
}
```

Tämän jälkeen otsikko ja hyötykuorma pitää base64Url-koodata. Internetistä löytyi kevyt avoimen lähdekoodin base64-koodauskirjasto, jolla pystyttiin helposti koodaamaan teksti base64-muotoon. Kirjasto ei valmiiksi tukenut base64Url-koodausta, joten tähän tarkoitukseen luotiin erikseen oma funktio, joka korvaa base64-kirjaston tuottaman koodin merkit + ja / merkeillä – ja _. Baseu64Url-koodauksen jälkeen otsikko ja hyötykuorma yhdistettiin toisiinsa pisteellä erotettuina, ja lopputulos näyttää seuraavanlaiselta:

```
eyAiYWxnIjogIkVTMjU2IiwgInR5cCI6ICJKV1QiIH0=.ewogICJhdWQiOi-
JteS1wcm9qZWN0IiwKICAiaWF0IjoxNTA5NjUwODAxLAogICJle-
HAiOjE1MDk2NTQ0MDEKfQo=
```

Seuraavaksi tehtävänä oli allekirjoittaa base64Url-merkkijono käyttäen yksityistä avainta ja micro-ecc-kirjaston allekirjoitusalgoritmia. Ennen allekirjoitusta merkkijono pitää tiivistää käyttäen SHA-256-tiivistealgoritmia. Nordic Semiconductorin kehitysympäristö tarjoaa valmiiksi SHA-256-tiivistekirjaston, jota käytettiin merkkijonon tiivistämiseen.

Jotta tiiviste voidaan allekirjoittaa micro-ecc-kirjaston algoritmilla, tarvitsee julkisen avaimen olla tietyssä muodossa. Kirjasto vaatii avaimen sijaitsevan uint8_t-taulukossa heksadesimaaleina. Asiakasyritykseltä saamani yksityinen avain oli pem-muodossa. Pem-muodossa oleva avain voidaan näyttää heksadesimaalimuodossa käyttäen Nordic Se-

miconductorin nrfutil-työkalua, joka voidaan asentaa Windows-ympäristöön. Nrfutil-työkaluun syötetään pem-tiedostomuodossa oleva yksityinen avain, ja tuloksena saadaan yksityinen avain heksadesimaalimuodossa. Saaduista heksadesimaaliarvoista poistetaan ensimmäinen heksadesimaali 0x04, joka on OpenSSL-ohjelman luoma etuliitetavu. Se kertoo, että avaimessa olevat ”pisteet”, joita käytetään elliptisten käyrien salausmenetelmissä, ovat pakkaamattomia.

Seuraavaksi pystyttiin allekirjoittamaan tiiviste käyttäen micro-ecc:n funktiota uECC_sign, joka ottaa vastaan parametreiksi yksityisen avaimen, tiivisteen, tiivisteen koon, 64 merkkiä pitkän uin8_t-tyyppisen taulukon, johon valmis allekirjoitus viedään, sekä allekirjoitukseen tarvittavan käyrän funktion. ES256:n käyttämä p-256-käyrä on micro-ecc-kirjastossa nimellä uECC_secp256r1.

Allekirjoituksen toimivuus voidaan varmistaa yksityisestä avaimesta luodulla julkisella avaimella. Julkinen avain generoitiin yksityisestä avaimesta käyttämällä nrfutil-ohjelmaa komennolla ”nrfutil keys display --key pk --format code c:\privatekey.pem”. Micro-ecc-kirjasto tarjoaa tähän uECC_verify()-tarkistusfunktion, joka ottaa vastaan parametreiksi julkisen avaimen, tiivisteen, tiivisteen koon, aiemmin luodun allekirjoituksen ja allekirjoitukseen käytettävän käyrän funktion. Funktio palauttaa onnistuessaan arvon 1 ja epäonnistuessaan arvon 0.

Ensimmäisellä yrityksellä funktio palautti arvon 0, joka kuvaa virhettä tarkistuksessa. Syyksi tähän paljastui OpenSSL-ohjelman luoma yksityinen avain, joka oli tavujärjestykseltään big endian -muodossa. Micro-ecc-kirjasto puolestaan oli käännetty ottavaksi vastaan little endian -tavujärjestyksessä olevia avaimia. Ongelma korjattiin asettamalla micro-ecc:n makefile-tiedostoon little endianin arvoksi 0 ja kääntämällä micro-ecc-kirjasto uudelleen. Ongelma olisi myös voitu korjata muuttamalla yksityisen avaimen tavujärjestys little endian -muotoon, mutta tämä olisi aiheuttanut ongelmia datankerääjän yrittäessä purkaa allekirjoitusta big endian -muodossa olevalla julkisella avaimella.

Allekirjoituksen varmistuttua toimivaksi (allekirjoitus voitiin purkaa avainparin julkisella avaimella ja vertailemalla tiivisteitä) voitiin allekirjoitus base64Url-koodata ja lisätä aiemmin luotujen base64Url-koodattujen otsikon ja hyötykuorman perään pisteellä erotettuna. Lopputulos näyttää seuraavanlaiselta:

eyAiYWxnIjo-gIkVTMjU2IiwgInR5cCI6ICJKV1QiIH0=.ewogICJhdWQiOi-JteS1wcm9qZWN0IiwKICAiaWF0IjoxNTA5NjUwODAxLAogICJle-HAiOjE1MDk2NTQ0MDEKfQo=.ZE2KfbPpiX3J454-YKd5VsgLSXCUqizAJot1A9WDiS-lIbTJZTCS3oysEu9mh6t3eg7H25imNQjb9cJvlK-I9Q

JWT:n toimivuus tarkistettiin jwt.io-verkkosivun työkalulla, johon JWT:n ja julkisen avaimen syöttämällä työkalu kertoo, voidaanko annetulla julkisella avaimella purkaa allekirjoitus ja vastaavatko puretun allekirjoituksen tiiviste viestin otsikosta ja väitteestä laskettua tiivistettä. JWT:n toimivuudesta sivusto kertoi signature verified -ilmoituksella.

Lisäksi haluttiin olla varmoja, että myös datankerääjä osaa purkaa luodun JWT:n. Kuten aiemmin mainitsin, Google tarjoaa esimerkkikoodeja MQTT-viestin ja JWT:n luomiseen. Esimerkkikoodi asennettiin Linux-ympäristöön, jossa esimerkkikoodin luoma JWT korvattiin omalla mikrokontrollerilla tuotetulla JWT:llä, joka lähetettiin MQTT-paketin salasanakentän mukana datankerääjän MQTT-välittäjälle. Datankerääjä sai MQTT-paketin vastaan ja pystyi todentamaan MQTT-paketin tulevan luotetusta lähteestä. Näin voitiin todeta mikrokontrollerilla luodun JWT:n toimivuus myös datankerääjällä.

Seuraavaksi yritettiin lähettää MQTT-pakettia Narrowband-verkon kautta datankerääjälle. Datankerääjän MQTT-välittäjä käyttää porttia 8883, joka on varattu TLS-salatulle yhteydelle. U-bloxin modeemissa on valmiina sisäänrakennettu TLS 1.2 -suojausprotokolla, jota päätettiin käyttää sen helppokäyttöisyyden vuoksi. TLS-yhteyden muodostamisen aikana kävi ilmi, että datankerääjä vaatii TLS-versio 1.2:n lisäksi suojauspaketin (engl. cipher suite), jota molempien osapuolten tulisi tukea. U-bloxin manuaalista kävi ilmi, että siinä ei ole datankerääjän vaatimaa suojauspakettia. U-bloxin tuesta kerrottiin, että päivitys modeemille on tulossa, ja se tukee kyseistä suojauspakettia. TLS-yhteyden olisi voinut luoda myös ohjelmallisesti itse, mutta aika sen toteuttamiseen ei riittänyt.

Lopputulos

Vaikka suojatun yhteyden luominen datankerääjälle ei onnistunut, saatiin paljon yhteyden muodostamiseen ja käyttäjän todentamiseen tarvittavia tekniikoita toteutettua. Lähitulevaisuudessa, kun u-bloxin modeemiin saadaan päivitys tukemaan datankerääjän vaatimaa suojauspakettia TLS-yhteyden luomiseen, on tarvittava todentamistekniikka jo luotu valmiiksi.

5 Yhteenveto

Insinöörityössä oli tavoitteena rakentaa ja ohjelmoida prototyyppi, joka lähettää turvallisesti tietoa IoT-laitteelta Googlen IoT Core -pilvipalveluun käyttäen tiedonsiirrossa uutta Narrowband IoT -verkkoteknologiaa. Tiedonsiirtoon tarvittavan suojauspaketin tuen puuttumisen kehitysalustasta ja työhön käytetyn ajan loppumisen vuoksi prototyyppi jäi osittain keskeneräiseksi. Työhön käytetystä kokonaisajasta suurin osa kului kirjastojen toimintakuntoon saattamiseen mikrokontrollerissa ja JSON Web Tokenin allekirjoituksen menetelmien työstämiseen.

Opinnäytetyötä tehdessä sain huomata, että kehitetyt ja turvalliset salaustekniikat vievät edelleen paljon laskentatehoa ja kuluttavat langattoman IoT-laitteen paristoa merkittävästi. Allekirjoitusmenetelmäksi JSON Web Tokeniin valittiin elliptisten käyrien allekirjoitusalgoritmi (ECDSA) sen keveyden ja lyhyiden avaimien ansiosta.

Tuloksena saatiin toimiva MQTT-tiedonsiirtoprotokolla, jolla luotiin onnistuneesti yhteys testipalvelimeen, elliptisten käyrien salausmenetelmiin käytettävän micro-ecc-kirjaston toiminta sekä JSON Web Tokenin luominen nRF52-kehitysalustalla. Narrowband IoT -verkkoteknologian kanssa ei haivattu ongelmia. Verkko toimi moitteettomasti koko projektin ajan, ja kuuluvuus oli hyvä Metropolian Leppävaaran-kampuksen kellarikerroksiin asti.

Huolimatta siitä, että tiedonsiirtoon vaadittavan salauspaketin puuttumisen vuoksi prototyyppiä ei voitu viimeistellä toimintakuntoon, olen työn toteutukseen ja lopputulokseen melko tyytyväinen. Muilta osin saatiin selvitettyä insinöörityön tavoitteiksi asetetut asiat, kuten Narrowband IoT -verkkoteknologian toimivuus tähän käyttötarkoitukseen.

Jatkokehitys

Prototyyppi on karkea malli, jolla pyrittiin demonstroimaan turvallinen tiedonsiirto ja ottamaan siihen käyttöön tarvittavat protokollat. Virhetilanteiden tarkistukset jäivät mikrokontrollerin ja modeemin välillä vähäisiksi, joten niitä pitäisi kehittää lisää lopputuotteeseen. Tällä hetkellä prototyypissä ainoastaan lähetetään MQTT-paketteja eikä tarkasteta, menikö paketti perille asti. Tähän pitäisi ottaa käyttöön MQTTClient-kirjasto, joka tarjoaa QoS (Quality of Service)-toiminnallisuudet, joilla voidaan tarkistaa yhteyden laatu MQTT-välittäjälle. Lisäksi pitäisi luoda aiheen tilaus MQTT-välittäjältä, jota voitaisiin esimerkiksi käyttää IoT-laitteen konfiguroimiseen haluttuun tilaan.

Asiakkaalle toteutettavassa IoT-laitteessa tietoturvaan täytyy tehdä prototyyppiin verrattuna muutamia muutoksia. Ne tiedostettiin jo työn alkuvaiheessa, mutta prototyyppiä varten niiden demonstroimiseen insinöörityöhön varattu aika ei riittänyt. Esimerkiksi yksityisen avaimen pitäminen mikrokontrollerin muistissa on aina riski. Lopullisessa tuotteessa yksityinen avain tulee olemaan erillisellä TPM (Trusted Platform Module)-turvapiirillä.

Lähteet

1 Verkkojen ja palvelujen tietoturva. 2015. Verkkoaineisto. Viestintävirasto. <www.viestintavirasto.fi/ohjausjavalvonta/tekninentoimivuusjatietoturva/tietoturva.html> Luettu 3.4.2018.

2 Yksityishenkilöiden 5 yleisintä tietoturvauhkaa ja 5 toimivaa ratkaisua. 2015. Verkkoaineisto. Viestintävirasto. <www.viestintavirasto.fi/kyberturvallisuus/tietoturvanyt/2017/01/ttn201701051436.html> Luettu 3.4.2018.

3 Lehtola, Sami. 2018. Uusi EU:n tietosuoja-asetus astuu voimaan 25.5.2018 – Ketä se koskee ja mitkä ovat sen keskeisimmät muutokset? Verkkoaineisto. <www.emce.fi/blog/uusi-eun-tietosuoja-asetus-astuu-voimaan-25-5-2018-keta-koskee-mitka-keskeisimmat-muutokset> Luettu 3.4.2018.

4 Tietoturva. Verkkoaineisto. Suomen internetopas. <www.internetopas.com/yleistietoa/tietoturva> Luettu 3.4.2018.

5 Tiesitkö, että 600 merimerkkiä on varustettu sim-korteilla? 2017. Verkkoaineisto. Telia. <www.telia.fi/yrityksille/artikkelit/artikkeli/iot-pitaa-merivaylat-turvallisina> Luettu 3.4.2018.

6 Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions). 2018. Verkkoaineisto. Statista. <www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide> Luettu 4.4.2018.

7 Yritysjohdon opas IoT:n ja teollisen internetin hyödyntämiseen. Verkkoaineisto. Quva. <quva.fi/site/attachments/yritysjohdon_opas_IoT_ja_teollisen_internetin_hyodyntamiseen.pdf> Luettu 4.4.2018.

8 Ota esineiden internetin suojauskeinot haltuun. 2016. Verkkoaineisto. Viestintävirasto. <www.viestintavirasto.fi/kyberturvallisuus/tietoturvanyt/2016/06/ttn201606210934.html> Luettu 4.4.2018.

9 Tenhunen, Jari. Vinkkejä esineiden internetin tietoturvaan. Verkkoaineisto. <www.itewiki.fi/julkaisu/vinkkeja-esineiden-internetin-tietoturvaan> Luettu 4.4.2018.

10 Viljanen, Vesa. Suojattu viestintä. Verkkoaineisto. <www.yksityisyydensuoja.fi/suojattu-viestintä> Luettu 12.4.2018.

11 Viljanen, Vesa. Tietojen salaaminen. Verkkoaineisto. <www.yksityisyydensuoja.fi/tietojen-salaaminen> Luettu 12.4.2018.

12 Yleistä salausmenetelmistä. 2018. Verkkoaineisto. Helsingin yliopisto. <www.helpdesk.it.helsinki.fi/ohjeet/tietoturva-ja-pilvipalvelut/tietoturva/yleista-salausmenetelmista> Luettu 12.4.2018.

13 Pietikäinen, Suvi. 2015. Tiedon salaaminen - tekninen viitekehys. Verkkoaineisto. <www.vahtiohje.fi/web/guest/tiedon-salaaminen-tekninen-viitekehys> Luettu 12.4.2018.

14 Ruishalme, Pirkka. DYTT Kryptaus. Verkkoaineisto. <www.ruipi.wikispaces.com/DYTT+Kryptaus> Luettu 12.4.2018.

15 Linna, Juho. 2005. Elliptisiin käyriin perustuvat kryptosysteemit. Verkkoaineisto. <www.math.tut.fi/julkaisut/pdf/di_juho_linna.pdf> Luettu 13.4.2018.

16 Nuutinen, Markku. Salaus. Verkkoaineisto. <www.askoik.kapsi.fi/koulu/SUORITETUT%20KURSSIT%204.%20vuosi/Tietoturvan%20perusteet/Luentomonisteet/TTP_5.pdf> Luettu 13.4.2018.

17 Symantec SSL Certification with the ECC Algorithm. Verkkoaineisto. Symantec. <www.symantec.com/content/en/us/enterprise/fact_sheets/b-symantec_ssl_certification_with_the_algorithm_DS.en-us.pdf> Luettu 13.4.2018.

18 Narrowband IoT (NB-IoT). Verkkoaineisto. Gemalto. <www.gemalto.com/m2m/development/innovation-technology/nb-iot> Luettu 8.4.2018.

19 DNA tuo NB-IoT-tekniikan käyttöön. 2017. Verkkoaineisto. Uusiteknologia.fi. <www.uusiteknologia.fi/2017/05/29/dna-tuo-nb-iot-tekniikan-kayttoon> Luettu 8.4.2018.

20 Glassman, Simon. 2017. Narrowband IoT – Solving the last-mile problem of the internet of things. Verkkoaineisto. <www.enterpriseiotinsights.com/20170828/channels/opinion/NB-IoT-last-mile-tag10> Luettu 8.4.2018.

21 Mouhamedou, Youssouf Ould Cheikh. 2018. Narrowband Internet of Things (NB-IoT) Overview. Verkkoaineisto. <www.grandmetric.com/2018/02/16/narrowband-internet-of-things-nb-iot-an-overview> Luettu 8.4.2018.

22 Narrowband IoT (NB-IoT). 2018. Verkkoaineisto. U-blox. <www.u-blox.com/en/narrowband-iot-nb-iot-0> Luettu 9.4.2018.

23 LTE Cat M1. 2018. Verkkoaineisto. U-blox. <www.u-blox.com/en/lte-cat-m1> Luettu 9.4.2018.

24 Frequently Asked Questions. Verkkoaineisto. Mqtt.org. <www.mqtt.org/faq>. Luettu 9.4.2018.

25 Yuan, Michael. 2017. Getting to know MQTT. Verkkoaineisto. <www.ibm.com/developerworks/library/iot-mqtt-why-good-for-iot/index.html> Luettu 9.4.2018.

26 Rouse, Margaret. MQTT (MQ Telemetry Transport). Verkkoaineisto. <www.internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport> Luettu 10.4.2018.

27 Introduction to JSON Web Tokens. Verkkoaineisto. Auth0. <www.jwt.io/introduction> Luettu 11.4.2018.

28 Debugger. Verkkoaineisto. Auth0. <www.jwt.io> Luettu 12.4.2018.

29 Allen, Brock. 2014. Base64url encoding. Verkkoaineisto. <www.brockallen.com/2014/10/17/base64url-encoding> Luettu 11.4.2018.

30 Base64 decode. Verkkoaineisto. Base64decode.net. <www.base64decode.net> Luettu 11.4.2018.

31 EVK-R4. Verkkoaineisto. U-blox. <www.u-blox.com/en/product/evk-r4> Luettu 13.4.2018.

32 nRF52840. Verkkoaineisto. Nordic Semiconductor. <www.nordicsemi.com/eng/Products/nRF52840> Luettu 12.4.2018.

33 Keim, Robert. 2016. Back to Basics: The Universal Asynchronous Receiver/Transmitter (UART). Verkkoaineisto. <www.allaboutcircuits.com/technical-articles/back-to-basics-the-universal-asynchronous-receiver-transmitter-uart> Luettu 15.4.2018.